Spectral learning with proper probabilities for finite state automation

Hadrien Glaude, Cyrille Enderli, Olivier Pietquin

SPECTRAL LEARNING WITH PROPER PROBABILITIES FOR FINITE STATE AUTOMATON

Hadrien Glaude [⋆†] *Cyrille Enderli* [†] *Olivier Pietquin* [⋆◇]

[†]Thales Airborne Systems, Elancourt, France
[⋆]Univ. Lille, CRIStAL, UMR 9189, SequeL Team, Villeneuve d'Ascq, France
[◇]Institut Universitaire de France (IUF)

ABSTRACT

Probabilistic Finite Automaton (PFA), Probabilistic Finite State Transducers (PFST) and Hidden Markov Models (HMM) are widely used in Automatic Speech Recognition (ASR), Text-to-Speech (TTS) systems and Part Of Speech (POS) tagging for language modeling. Traditionally, unsupervised learning of these latent variable models is done by Expectation-Maximization (EM)-like algorithms, as the Baum-Welch algorithm. In a recent alternative line of work, learning algorithms based on spectral properties of some low order moments matrices or tensors were proposed. In comparison to EM, they are orders of magnitude faster and come with theoretical convergence guarantees. However, returned models are not ensured to compute proper distributions. They often return negative values that do not sum to one, limiting their applicability and preventing them to serve as an initialization to EM-like algorithms. In this paper, we propose a new spectral algorithm able to learn a large range of models constrained to return proper distributions. We assess its performances on synthetic problems from the PAutomaC challenge and real datasets extracted from Wikipedia. Experiments show that it outperforms previous spectral approaches as well as the Baum-Welch algorithm with random restarts, in addition to serve as an efficient initialization step to EM-like algorithms.

Index Terms— Baum-welch, learning automata, spectral learning, language models, non-negative matrix factorization

1. INTRODUCTION

Natural language processing (NLP) makes extensive use of automata and relies on training data to learn their parameters. The focus of this paper is a novel algorithm for learning parameters of a general family of automata, capturing most of the ones that are being used for NLP, *i.e.* Probabilistic (non-deterministic) Finite Automata (PFA). A PFA is a generative model for sequential data. Intuitively a PFA is similar to a Hidden Markov Model (HMM) in that it consists of a set of states, each of which when visited emits a symbol according to an emission probability distribution. Probabilistic Finite State Transducers (PFST) are also similar to PFA defined over pairs of input and output symbols. These models have been used, for example, to train POS tagging models [1], Automatic Speech Recognition [2] and other applications [3, 4]. Learning such automata parameters often rely on Expectation-Maximization (EM)-like algorithms trying to maximize the joint likelihood like Baum-Welch, Viterbi, Variational Bayes or Gibbs Sampling. However, these algorithms are iterative procedures and converge to local minima. In addition to being prone to get stuck into local optima, these algorithms are computationally expensive, to the point where obtaining good solutions for large models becomes intractable.

A recent alternative line of work consists in designing learning algorithms for latent variable models exploiting the so-called Method of Moments (MoM). The MoM leverages the fact that low order moments of distributions contain most of the distribution information and are typically easy to estimate. The MoM have several pros over iterative methods: it can be applied to a large variety of models [5, 6, 7, 8] and MoM-based algorithms are extremely fast as estimated moments can be computed in a linear time w.r.t. the number of samples. In addition, they are often consistent with theoretical guarantees in form of finite-sample bounds. This is particularly appealing because the resulting parameters can be used to initialize EM-like algorithms in a two-stage learning procedure, resulting in the best of both worlds: parameters efficiently computed by a MoM-based algorithm provide a good initialization for a maximum likelihood approach.

Beyond all the appealing traits of MoM-based algorithms, a well-known concern is the Negative Probability Problem (NPP) [7]. When learnt from samples, MoM estimators are not guaranteed to find a valid set of parameters. Although errors in the estimated parameters are bounded, the parameters themselves may lie outside the class of models that define proper distributions. NPP prevents the use of MoM-based algorithms to initialize a local search algorithm like EM. As mentioned in [9, 10], this is a longstanding issue. Most of the recent attempts try to project the learnt model onto the space of valid model parameters [10]. While these heuristics produce a usable model, the resulting model may no longer be close to the true parameters. In this paper, we adopt a different approach by identifying a subclass of models realizing proper distributions and learnable through the MoM. Although this subclass does not encompass all the generative models for sequential data, it has an expressiveness large enough to closely approximate many of the distributions used in practice. In addition, learnt models from that subclass can serve as a good initialization to EM-like algorithms. The paper is organized as follows: in Section 2, we recall the definition of a PFA and develop the basic Spectral learning algorithm; in Section 3 we define a particular form of automaton capturing proper distributions and a polynomial-time learning algorithm that we call SepPFA (for Separable PFA); Section 4 analyses classes of automaton that can and cannot be learnt by SepPFA; finally, we assess the performance of SepPFA on synthetic problems and real large datasets that cannot be handled by traditional methods like EM.

2. BACKGROUND

2.1. Probabilistic Finite Automaton

PFA are graphical models constrained to represent distributions over sequences of symbols. Let Σ be a set of symbols, also called an al-

phabet. We denote by $\Sigma^\star$, the set of all finite words made of symbols of Σ, including the empty word ε. Words of length k form the set Σ^k. Let u and $v \in \Sigma^\star$, uv is the concatenation of the two words and $u\Sigma^\star$ is the set of finite words starting by u. We are interested in capturing a distribution over $\Sigma^\star$. Let p be such a distribution, for a set of words $\mathcal{S}$, we define $p(\mathcal{S}) = \sum_{u\in\mathcal{S}} p(u)$, in particular we have that $p(\Sigma^\star) = 1$. In addition, for any word u we define $\overline{p}$ such that $\overline{p}(u) = p(u\Sigma^\star)$. Thus, $\overline{p}$ defines distributions over prefixes of fix length : $\forall n$, $\sum_{u\in\Sigma^n} \overline{p}(u) = 1$.

Some of these distributions can be modeled by graphical models called Probabilistic Finite Automaton (PFA).

Definition. *A PFA is a tuple $\langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$, where Σ is an alphabet and Q is a finite set of states. Matrices $A_o \in \mathbb{R}^{+|Q|\times|Q|}$ contain the transition weights. The vectors $\boldsymbol{\alpha}_\infty \in \mathbb{R}^{+|Q|}$ and $\boldsymbol{\alpha}_0 \in \mathbb{R}^{+|Q|}$ contain respectively the terminal and initial weights. These weights should verify,*

$$\mathbf{1}^\top \boldsymbol{\alpha}_0 = 1 \qquad \boldsymbol{\alpha}_\infty + \sum_{o\in\Sigma} A_o \mathbf{1} = \mathbf{1} \tag{1}$$

A PFA realizes a distribution over $\Sigma^\star$ defined by

$$p(u) = p(o_1 \dots o_k) = \boldsymbol{\alpha}_0^\top A_u \boldsymbol{\alpha}_0 = \boldsymbol{\alpha}_0^\top A_{o_1} \dots A_{o_k} \boldsymbol{\alpha}_\infty. \tag{2}$$

Because of the constraints defined in Equation (1), the weights belong to $[0, 1]$ and can be viewed as probabilities over initial states, terminal states and transitions with symbol emission. For a word, we define a path as a sequence of states starting in an initial state, transiting from state to state, emitting symbols of the word in each state and exiting in a final state. The probability of a path is the product of the weights along the path including initial and final weights. Hence, the probability of a word is given by the sum of all paths probabilities, as written in Equation (2). A path with a positive probability is called an accepting path.

PFA define a particular kind of Multiplicity Automaton (MA). A MA is also a tuple $\langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ but without any constraints on the weights, which can be negative and lose their probabilistic meaning. We call a formal power series a function that associates to any word of $\Sigma^\star$ a real. Thus, MA can realize formal power series that are not distributions. A MA that realizes a distribution is called a Stochastic MA (SMA). SMA are more general than PFA because they do not require the non-negativity of the weights. The Spectral algorithm presented in the next section relies on the Hankel matrix representation of a series to learn MA.

2.2. Spectral Learning

Let $p : \Sigma^\star \to \mathbb{R}$ be a formal power series, we define $H \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$ the bi-infinite Hankel matrix whose rows and columns are indexed by $\Sigma^\star$ such that $H[u, v] = p(uv)$,

$$H = \begin{array}{c} \\ \varepsilon \\ a \\ b \\ aa \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \varepsilon & a & b & aa & \dots \end{array} \\ \begin{pmatrix} p(\varepsilon) & p(a) & p(b) & p(aa) & \dots \\ p(a) & p(aa) & p(ab) & p(aaa) & \dots \\ p(b) & p(ba) & p(bb) & p(baa) & \dots \\ p(aa) & p(aaa) & p(aab) & p(aaaa) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

When p is a distribution, H contains occurrence probabilities that can be estimated from samples by empirical frequencies. Let for all $o \in \Sigma$, $H_o \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$, $\mathbf{h}_S \in \mathbb{R}^{\Sigma^\star}$ and $\mathbf{h}_P \in \mathbb{R}^{\Sigma^\star}$ be such that $H_o(u, v) = p(uov)$, $\mathbf{h}_S(u) = \mathbf{h}_P(u) = p(u)$. These vectors and matrices can be extracted from H. The Hankel representation of formal series lies at the heart of all MoM-based learning algorithms, because of the following fundamental theorem.

Theorem 1 (See [11])**.** *Let p be a formal power series realized by a MA with n states, then $\mathrm{rank}(H_p) \leq n$. Conversely, if the Hankel matrix H_p of a formal power series p has a finite rank n, then p can be realized by a MA with exactly n states but not less.*

For a MA with n states, observe that $H[u, v] = (\boldsymbol{\alpha}_0^\top A_u)(A_v \boldsymbol{\alpha}_\infty)$. Let $P \in K^{\Sigma^\star\times n}$ and $S \in K^{n\times\Sigma^\star}$ be matrices defined as follows,

$$P = ((\boldsymbol{\alpha}_0^\top A_u)^\top)_{u\in\Sigma^\star}^\top, \qquad S = (A_v \boldsymbol{\alpha}_\infty)_{v\in\Sigma^\star},$$

then $H = PS$. Moreover, we have that,

$$H_o = P A_o S, \quad \mathbf{h}_S^\top = \boldsymbol{\alpha}_0^\top S, \quad \mathbf{h}_P = P\boldsymbol{\alpha}_\infty. \tag{3}$$

So the MA parameters can be recovered by solving Equation (3). Hopefully, we do not need to consider the bi-infinite Hankel matrix to recover the underlying MA. Given a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ of prefixes and suffixes, we denote by $H_\mathcal{B}$ the sub-block of H. Similarly, $H_{\mathcal{B},o}$ is a sub-block of H_o. A basis $\mathcal{B}$ is *complete* if $H_\mathcal{B}$ has the same rank than H. In [12], the author shows that if $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a complete basis, by defining P over $\mathcal{P}$, S over $\mathcal{S}$ and H, H_o over $\mathcal{B}$, we can recover a MA using Equation (3). In the literature, several methods are used to build a complete basis from data. For example, one can use all prefixes and suffixes that appear in the training set.

Once a basis is chosen, the Spectral algorithm first estimates the probabilities in $\hat{H}_\mathcal{B}$ by empirical counts. Then, it recovers a factorized form of $\hat{H}_\mathcal{B} = \hat{U}\hat{D}\hat{V}^\top$ through a truncated Singular Value Decomposition (SVD). Finally, setting $\hat{P} = \hat{U}\hat{D}$ and $\hat{S} = \hat{V}^\top$, the algorithm solves Equation (3),

$$\begin{aligned} \hat{A}_o &= \hat{D}^{-1}\hat{U}^\top \hat{H}_{\mathcal{B},o}\hat{V}\,, \\ \hat{\boldsymbol{\alpha}}_0^\top &= \hat{\mathbf{h}}_S^\top \hat{V}\,, \\ \hat{\boldsymbol{\alpha}}_\infty &= \hat{D}^{-1}\hat{U}^\top \hat{\mathbf{h}}_P\,. \end{aligned}$$

In the experiments, following the advice of [13], we normalized the feature-variance of the coefficients of the Hankel matrix by independently scaling each row and column by a factor $c_u = \sqrt{|\mathcal{S}|/(\#u + 5)}$, where $\#u$ is the number of occurrences of u. In addition, depending on the problem, it can be better to work with other series derived from p. For example, the substring-based series $p^{substring}(u) = \sum_{w,v\in\Sigma^\star} p(wuv)$ is related to p^{string}. According to [12], if p^{string} is realized by a SMA, then $p^{substring}$ is too. In addition, [12] provides an explicit conversion between string-based SMA and substring-based SMA preserving the number of states. For all MoM-based algorithms compared on the PAutomaC, we used the series leading to the best results for each problem.

Although the Spectral algorithm can return a MA arbitrary close to a SMA or a PFA that realizes the target distribution, it does not ensure that the returned MA will be a SMA or a PFA. This causes the so-called Negative Probability Problem and prevents to use the learned MA as an initialization to EM. Indeed, when the Spectral algorithm computes a low rank approximation of the sampled Hankel matrix, it does not preserve the Hankel structure. In [10], authors proposed to compute a Structured Low-Rank Approximation (SLRA). However, the problem is strongly non-convex and thus cannot be applied directly on large scale problems.

3. RESIDUAL REPRESENTATION

This section defines a particular representation of Multiplicity Automaton called residual realizing a formal power series p. As in the Spectral algorithm, our method assumes that a complete basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is given as input. Let $\mathcal{P}'$ be a subset of $\mathcal{P}$, such that $\text{rank}(H_{\mathcal{B}}) = \text{rank}(H_{\mathcal{B}'})$ where $\mathcal{B}' = (\mathcal{P}', \mathcal{S})$. In addition, let's assume there exists for all $u \in \mathcal{P}'$ and all $o \in \Sigma$, two sets of variables $\{a^v_{u,o}\}_{v\in\mathcal{P}'}$ and $\{a^v_\varepsilon\}_{v\in\mathcal{P}'}$ verifying, for all $w \in \Sigma^\star$,

$$\frac{p(uow)}{\overline{p}(uo)} = \sum_{v\in\mathcal{P}'} a^v_{u,o}\frac{p(vw)}{\overline{p}(v)} \text{ and } p(w) = \sum_{v\in\mathcal{P}'} a^v_\varepsilon \frac{p(vw)}{\overline{p}(v)} \quad (4)$$

In $\mathbb{R}$-MA, states are associated to elements of $\mathcal{P}'$ and the weights are defined as follows,

$$\boldsymbol{\alpha}_0^\top = (a^u_\varepsilon)^\top_{u\in\mathcal{P}'} \quad (5)$$

$$\boldsymbol{\alpha}_\infty = \left(\frac{p(u)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'} \quad (6)$$

$$\forall u, v \in \mathcal{P}',\ A_o[u,v] = a^v_{u,o}\frac{\overline{p}(uo)}{\overline{p}(u)}. \quad (7)$$

The next proposition gives a necessary condition for a formal power series p to admit a residual representation.

Proposition 2. *If there exists $\mathcal{P}'$ and $\{a^v_{u,o}\}_{v\in\mathcal{P}'}$, $\{a^v_\varepsilon\}_{v\in\mathcal{P}'}$ verifying Equation* (4) *for all $w \in \Sigma^\star$, then for any word $w = o_1...o_n$, we have,*

$$A_w\boldsymbol{\alpha}_\infty = A_{o_1}...A_{o_n}\boldsymbol{\alpha}_\infty = \left(\frac{p(uw)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'}, \quad (8)$$

and so,

$$p(w) = \boldsymbol{\alpha}_0^\top A_w \boldsymbol{\alpha}_\infty.$$

Proof. The proof is done by induction on $|w|$. For the empty word, Equation (8) follows from the definition of $\boldsymbol{\alpha}_\infty$. Now, for $w = o_1...o_n$, the induction hypothesis gives,

$$A_{o_2}...A_{o_n}\boldsymbol{\alpha}_\infty = \left(\frac{p(uo_2...o_n)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'},$$

So we have,

$$\begin{aligned} A_w\boldsymbol{\alpha}_\infty &= \left(\sum_{v\in\mathcal{P}}\frac{p(vo_2...o_n)}{\overline{p}(v)} \times a^v_{u,o_1}\frac{\overline{p}(uo)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'} \\ &= \left(\frac{\overline{p}(uo)}{\overline{p}(u)}\sum_{v\in\mathcal{P}}\frac{p(vo_2...o_n)}{\overline{p}(v)}a^v_{u,o_1}\right)_{u\in\mathcal{P}'} \\ &= \left(\frac{p(uo_1o_2...o_n))}{\overline{p}(u)}\right)_{u\in\mathcal{P}} = \left(\frac{p(uw)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'}, \end{aligned}$$

where the last step uses Equation (4), with $w = o_2...o_n$. In addition, Equation (4) gives,

$$\boldsymbol{\alpha}_0^\top A_w\boldsymbol{\alpha}_\infty = \sum_{u\in\mathcal{P}'} a^u_\varepsilon\frac{p(uw)}{\overline{p}(u)} = p(w)$$

□

Note that, for a MA with k states, the infinite set of linear equations given by Equation (4) for all $w \in \Sigma^\star$ forms a system of rank k and admits a solution which can be found by solving the set of Equation (4) for all $w \in \mathcal{S}$, if $\mathcal{B}'$ is complete.

There exists many methods [7, 10] to compute a minimal subset $\mathcal{P}'$ such that $\mathcal{B}'$ is complete. Once $\mathcal{B}'$ is found, one has to solve Equation (4) by linear regression using empirical estimations $\hat{p}$ of p, and then compute the weights using $\hat{p}$ and the $a^v_{u,o}$s.

Recalling that our goal is to learn proper distributions, one would like to add constraints to ensure that the residual form MA learned by regression realizes a proper distribution. Unfortunately, this would require two things : 1) the non-negativity of series for any word ($\forall \in \Sigma^\star$, $p(u) \geq 0$), 2) the convergence of the series to one ($\sum_{u\in\Sigma^\star} p(u) = 1$). Although 2) can be checked in polynomial time, 1) requires adding an infinite set of constraints during the linear regression step. That is why, in general, verifying if a MA is a SMA is undecidable [14]. So, we restrict ourselves to the learning of PFA in their residual form.

3.1. Learning Residual PFA

In this section, we detail how to build a residual representation of a PFA that realizes a distribution (under some conditions on this distribution). As in the previous section, we start with a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$. In contrast to SMA, for PFA the non-negativity of the series is ensured by the non-negativity of the initial, final and transition weights. In addition, the convergence of the series is shown in [14] to be equivalent to the finite set of linear constraints given in Equation (1). Thus, constraints will be added to find both the $a^v_{u,o}$s and a minimal subset $\mathcal{P}'$ of $\mathcal{P}$, that ensures the non-negativity and the convergence of the series.

Firstly, the convergence of a series is shown in [14] to be equivalent to the finite set of linear constraints given in Equation (1). Thus, in its residual form, a PFA must verifies for all $u \in \mathcal{P}'$ that

$$\frac{p(u)}{\overline{p}(u)} + \sum_{\substack{v\in\mathcal{P}\\ o\in\Sigma}} a^v_{u,o}\frac{\overline{p}(uo)}{\overline{p}(u)} = 1 \text{ and } \sum_{v\in\mathcal{P}} a^v_\varepsilon = 1. \quad (9)$$

Note that, when working with the true distribution p, Equation (9) is always satisfied, if Equation (4) holds for all $w \in \Sigma^\star$. Indeed, we have

$$p(u) + \sum_{\substack{v\in\mathcal{P}\\ o\in\Sigma}} a^v_{u,o}\overline{p}(uo) = p(u) + \sum_{o\in\Sigma}\overline{p}(uo) = \overline{p}(u),$$

because, summing Equation (4) over $w \in \Sigma^\star$ gives $\sum_{v\in\mathcal{P}} a^v_{u,o} = 1$ and $\sum_{v\in\mathcal{P}} a^v_\varepsilon = 1$. However, if p and $\overline{p}$ are estimated, to ensure the convergence of the series, the constraints defined in Equation (9) has to be added when solving Equation (4).

Secondly, as the non-negativity of the weights in the residual form of a PFA is equivalent to the non-negativity of the $a^v_{u,o}$s and a^v_εs, we must find a subset $\mathcal{P}'$ verifying for all $w \in \Sigma^\star$ Equation (4) where $a^v_{u,o} \geq 0$ and $a^v_\varepsilon \geq 0$. This constraint can be rewritten in a vector form where $\mathbf{p}_u$ is the infinite vector $\left(\frac{p(uw)}{\overline{p}(u)}\right)_{w\in\Sigma^\star}$,

$$\begin{aligned} \forall u \in \mathcal{P}',\ o \in \Sigma,\ \mathbf{p}_{uo} &= \sum_{v\in\mathcal{P}'} a^v_{u,o}\mathbf{p}_v \text{ with } a^v_{u,o} \geq 0 \text{ and} \\ \mathbf{p}_\varepsilon &= \sum_{v\in\mathcal{P}'} a^v_\varepsilon\mathbf{p}_v \text{ with } a^v_{u,o} \geq 0\,. \end{aligned}$$

meaning that

$$\begin{aligned} \forall u \in \mathcal{P}',\ o \in \Sigma,\ \mathbf{p}_{uo} &\in \text{coni}\{\mathbf{p}_u \mid u \in \mathcal{P}'\} \text{ and} \\ \mathbf{p}_\varepsilon &\in \text{coni}\{\mathbf{p}_u \mid u \in \mathcal{P}'\}\,. \end{aligned}$$

where coni denote the conical hull of a set. Finding a subset $\mathcal{P}'$ like that is not trivial in the general case. As an approximation, we propose to assume the following condition :

Condition 1 (k-Separability)**.** *There are k prefixes $u_1, ..., u_k$ of $\mathcal{P}$ such that*

$$\text{coni}\{\mathbf{p}_u \mid u \in \Sigma^\star\} = \text{coni}\{\mathbf{p}_{u_1}, ...\mathbf{p}_{u_k}\}, \text{ and}$$
$$\text{coni}\{\mathbf{p}_u^{\mathcal{S}} \mid u \in \mathcal{P}\} = \text{coni}\{\mathbf{p}_{u_1}^{\mathcal{S}}, ...\mathbf{p}_{u_k}^{\mathcal{S}}\},$$

where, $\mathbf{p}_u^{\mathcal{S}}$ is the restriction of $\mathbf{p}_u$ on the basis $\mathcal{S}$.

Section 4 details the models satisfying this condition. Now, assuming the k-separability condition, let $\mathcal{P}'$ be $\{u_1, ..., u_k\}$, then Equation (4) holds trivially for all $w \in \Sigma^\star$.

Algorithm 1 The SepPFA algorithm

Input: A basis $\mathcal{B}$ satisfying the k-separability condition.
Output: A PFA with a residual form $\langle \Sigma, \mathcal{P}', \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$.

1: Build empirical estimates of p and $\overline{p}$.
2: Find the k edges $\{\mathbf{p}_{u_1}^{\mathcal{S}}, ..., \mathbf{p}_{u_k}^{\mathcal{S}}\}$ of $\text{coni}\{\mathbf{p}_u^{\mathcal{S}} \mid u \in \mathcal{P}\}$
3: $\mathcal{P}' \leftarrow \{ u_1, ..., u_k\}$
4: Solve the following optimization problem

$$\underset{\{a_{u,o}^v\}}{\text{argmin}} \sum_{u,o} \left\| \mathbf{p}_{uo}^{\mathcal{S}} - \sum_{v\in\mathcal{P}'} a_{u,o}^v \mathbf{p}_v^{\mathcal{S}} \right\|_2$$
$$\text{s.t.} \sum_{v\in\mathcal{P}, o\in\Sigma} a_{u,o}^v \overline{p}(uo) = \overline{p}(u) - p(u) \text{ and } a_{u,o}^v \geq 0\,.$$

5: Solve the following optimization problem

$$\underset{\{a_\varepsilon^v\}}{\text{argmin}} \left\| \mathbf{p}_\varepsilon^{\mathcal{S}} - \sum_{v\in\mathcal{P}'} a_\varepsilon^v \mathbf{p}_v^{\mathcal{S}} \right\|_2 \text{ s.t. } \sum_{v\in\mathcal{P}} a_\varepsilon^v = 1 \text{ and } a_{u,o}^v \geq 0.$$

6: $\boldsymbol{\alpha}_0^\top \leftarrow (a_\varepsilon^u)_{u\in\mathcal{P}'}^\top$
7: $\boldsymbol{\alpha}_\infty \leftarrow \left(\frac{p(u)}{\overline{p}(u)}\right)_{u\in\mathcal{P}'}$
8: For all $u, v \in \mathcal{P}'$, $A_o[u, v] \leftarrow a_{u,o}^v \frac{\overline{p}(uo)}{\overline{p}(u)}$

We now detail Algorithm 1. In the second step, finding the k edges can be cast as a well-known problem: near-separable Non-negative Matrix Factorization (NMF). In the literature, many algorithms for NMF have been proposed. Although in its general form NMF is NP-Hard and ill-posed [15], in the near-separable case the solution is unique and can be found in a time proportional to $O(k\,|\mathcal{S}|\,|\mathcal{P}|)$. State-of-the-art algorithms for near-separable NMF comes with convergence guarantees and robustness analysis, making the near-separable assumption appealing. For Algorithm 1, we used the Successive Projection Algorithm (SPA) with a post-processing step described and analyzed in [16]. As rescaling the $\mathbf{p}_{u_i}^{\mathcal{S}}$s does not change the solution, we multiplied them by $\overline{p}(u_i)$ in order to normalize the estimation noise, before applying the SPA. Indeed, $\overline{p}(u_i)$ is an approximation of the inverse of the variance.

In the fourth and fifth steps of Algorithm 1 the minimization problem is actually an instance of a quadratic optimization problem under linear constraints, that can be solved in polynomial time using a solver like *MOSEK*. As the problem is strongly convex, it admits a unique stationary point to which the solver converges.

Finally, if the target distribution p and the given basis $\mathcal{B}$ satisfy the k-separability assumption, Algorithm 1 returns a PFA with k states in a residual form that realizes p. This automaton can then serve as an initialization to a Baum-Welch algorithm.

4. THE SEPARABILITY CONDITION

After defining the family of SPA that can be estimated by Algorithm 1 given a suitable basis, we show that k-separable PFA encompass a large variety of models, including the Residual PFA (RPFA) defined in [17]. We start with few definitions.

Definition. *A k-separable PFA is a PFA such that, there exists k prefixes $u_1, ..., u_k$ satisfying*

$$\text{coni}\{\mathbf{p}_u \mid u \in \Sigma^\star\} = \text{coni}\{\mathbf{p}_{u_1}, ...\mathbf{p}_{u_k}\}.$$

Thus, if p is realized by a k-separable PFA $\mathcal{M}$ and a suitable basis $\mathcal{B}$ is given, Algorithm 1 launched with parameters k and $\mathcal{B}$ is able to recover the residual form of $\mathcal{M}$ from the true values of p. Next, we show that RPFA with k states are k-separable. First, we need few definitions. Let $\mathcal{M} = \langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ be a PFA, for each state $q \in Q$ we define $\mathcal{M}_q = \langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\beta}_0, \boldsymbol{\alpha}_\infty \rangle$ be the PFA such that $\boldsymbol{\beta}_0 = \mathbb{1}_q$. We denote by p_q the distribution realized by $\mathcal{M}_q$.

Definition. *Let $\mathcal{M} = \langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ be a PFA, then $\mathcal{M}$ is a Residual PFA (RPFA) iff*

$$\forall q \in Q,\, \exists u_q \in \Sigma^\star,\, \forall w \in \Sigma^\star p_q(w) = \frac{p(u_q w)}{\overline{p}(u_q)}.$$

Theorem 3. *A RPFA with k states is k-separable.*

Proof. Let $\mathcal{M} = \langle \Sigma, Q, \{A_o\}_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ be a RPFA. Let $\mathbf{q}_w$ be a vector such that $\mathbf{q}_w[q] = p_q(w) = \frac{p(u_q w)}{\overline{p}(u_q)}$. We denote by b_q the q-th coefficient of the vector $\boldsymbol{\alpha}_0^\top A_u / \overline{p}(u)$. Note that we have $b_q \geq 0$. By definition of RPFA, we have for all $u, w \in \Sigma^\star$ that,

$$\frac{p(uw)}{\overline{p}(u)} = \frac{1}{\overline{p}(u)} \boldsymbol{\alpha}_0^\top A_u \mathbf{q}_w = \sum_{q\in Q} b_q \frac{p(u_q w)}{\overline{p}(u_q)}$$

In terms of vectors, we have

$$\mathbf{p}_u = \sum_{q\in Q} b_q \mathbf{p}_{u_q} \text{ with } b_q \geq 0$$

□

The expressiveness of RPFA has been thoroughly studied in [17]. Authors have shown that all distributions realized by a PDFA can be realized by a RPFA but there are some distributions a RPFA can realized that cannot be realized by PDFA. Thus, RPFA strictly generalize PDFA. In [18], authors have shown that PDFA strictly generalize n-th order Markov chains for any n. On the bad side, PFA are strictly more general than RPFA [17]. In [19], authors gave explicit conversions between HMMs and PFA, showing that a PFA can represent an HMM with the same number of states and a HMM can represent a PFA using an increased number of states. Thus, PFA and HMM have the same expressiveness, although PFAs seems slightly more compact.

5. EXPERIMENTAL RESULTS

5.1. PAutomaC Challenge

The Probabilistic Automata learning Competition (PAutomaC) deals with the problem of learning probabilistic distributions from strings drawn from finite-state automata. From the 48 problems available, we have selected the same twelve problems than in [9], to provide a fair comparison with other algorithms. The generating model can be of three kinds: PFA, HMMs or PDFA. Four models have been selected from each class. A detailed description of each problem can be found in [20]. Table 1 compares SepPFA, BW-SepPFA (Baum-Welch initialized with SepPFA) and BW (with random restarts initialization) to best results (denoted by MoM) among the following MoM-based algorithms : CO [21] using strings statistics, Tensor [5] using strings statistics and Spectral using strings and substrings statistics. A description and comparison of these algorithms can be found in [9]. For SepPFA we used 500 prefixes and suffixes. The quality of a model can be measured by the quality of the probability distribution it realizes. The objective is to learn a MA realizing a series p close to the distribution $p_\star$, which generated the training set $\mathcal{T}$. The quality of p is measured by the perplexity that corresponds to the average number of bits needed to represent a word using the optimal code given by $p_\star$.

$$\text{Perplexity}(\mathcal{M}) = 2^{-\sum_{u\in\mathcal{T}} p_\star(u)\log(p_\mathcal{M}(u))}$$

	ID	BW-Sep	Sep	BW	MoM	True model
HMM	1	**38.68**(5)	68.91	500.10	44.77	29.90(63)
	14	116.92(15)	125.09	**116.84**	128.53	116.79(15)
	33	**32.11**(20)	38.44	32.14	49.22	31.87(13)
	45	**25.92**(5)	28.73	107.75	31.87	24.04(14)
PDFA	6	**67.09**(40)	83.71	67.32	95.12	66.98(19)
	7	51.30(15)	66.51	**51.27**	62.74	51.22(12)
	27	**59.39**(20)	74.48	94.40	102.85	42.43(19)
	42	**20.47**(35)	18.13	168.52	23.91	16.00(6)
PFA	29	**24.71**(30)	275.21	25.09	34.57	24.03(36)
	39	14.30(5)	18.96	**10.43**	11.24	10.00(6)
	43	**36.08**(5)	34.93	461.23	36.61	32.64(67)
	46	12.07(30)	105.62	**12.02**	25.28	11.98(19)

Table 1. Comparison with other algorithms for perplexity. Model sizes are listed in parentheses.

A grid search was performed to find the optimal rank for each of the performance metric. For each problem, the best between BW-SepPFA and SepPFA is indicated by a bold number and the best score between SepPFA and the best of other MoM-based algorithms is underlined. The score and the size of the true model is reported for comparison. First, SepPFA achieves the better score than MoM on the majority of the problems showing that RPFA can provide very good approximations of more complex models. When using BW on this dataset, without any stopping conditions, lot of over-fitting was observed. This explains why few iterations of BW degrades the performances of SepPFA on problem 42 and 43. The results in Table 1 have been obtained by stopping the BW algorithm when it does not improve the perplexity anymore. BW-SepPFA in comparison to BW achieves a lowest perplexity for 8 problems among 12. For the other 4 problems, the perplexity of both BW-SepPFA and BW are very close to the true one. This clearly argues for the use of Algorithm 1 as an initialization step to BW.

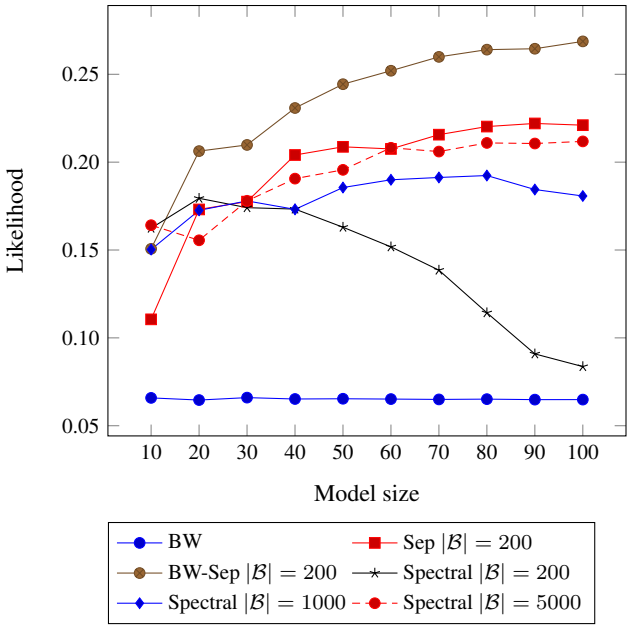

Fig. 1. Performances on the small Wikipedia training set.

5.2. Wikipedia

We evaluate Algorithm 1, on a large corpus of real-world Wikipedia text treated as chunked of 250 characters randomly extracted from the 2GB corpus used in [22]. Each character stands for an observation. There are 85 different characters in the data set. For the training phase, we used a small database of 500 chunks and a medium one of 50000 characters. For testing, an independent data set of 5000 chunks is used. We evaluate three algorithms : Baum-Welch (BW) with 5 random restarts of 4 iterations (then the best run is continued for 20 iterations), SepPFA (Sep) and Baum-Welch initialized with SepPFA (BW-Sep) with 20 iterations. Performance is measured by the average likelihood of the next observation given the past. More precisely, on chunk of 250 characters, the 50 firsts serve to initialize the belief, then the next observation is predicted based on the characters observed so far. After predicting the next observation, this is added to the past and a prediction is made for the next one until the end of the chunk is reached. The likelihood of all these predictions is then averaged over observations and chunks.

Many conclusions can be drawn from Figures 1 and 2. First, on the two datasets, Baum-Welch initialized using SepPFA performs best. Although BW and BW-SepPFA achieve almost the same likelihood on the joint distribution in our experiments, we can see on figs. 1 and 2 that BW achieves a very small likelihood on the conditional distribution. This means that the probabilities of rare events are incorrectly estimated by BW, where MoM-based algorithms are much more efficient. Actually, BW tends to over-fit the distribution, but as RPFA defines a simpler class of model it brings generalization properties and so is a good model to initialized BW. Comparing SepPFA and Spectral, SepPFA seems to have better generalization properties on small datasets, whereas Spectral is able to learn more complex models by better fitting the target distribution when more data is available. This can be explained by the fact that SepPFA is

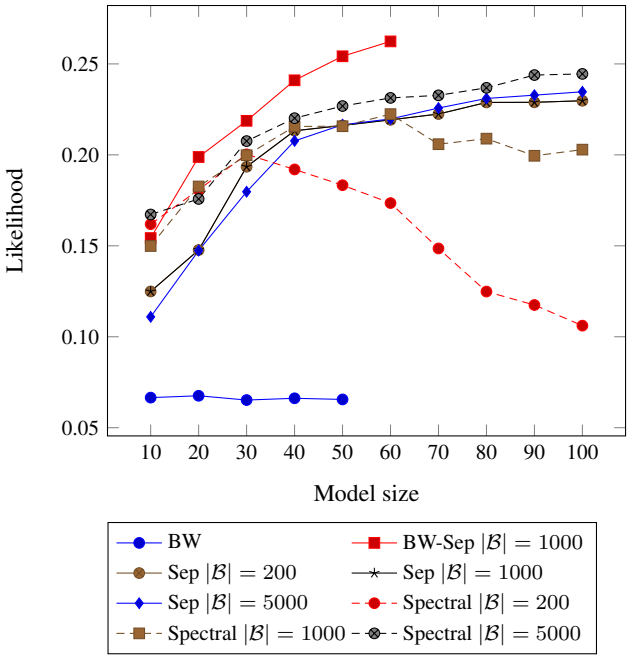

Fig. 2. Performances on the medium Wikipedia training set. For BW and BW-SepPFA, the computation was too long to be evaluated for all model sizes.

able to learn only a particular constrained subclass of MA. About the size of the basis, it denotes the number of prefixes or suffixes we used. In our experiments, we used the same set for prefixes and suffixes. This set contains the most frequents substrings of length up to 3 in the training set. SepPFA behaves better on small basis, whereas Spectral uses very large basis to capture the maximum of information. Indeed, the larger the basis is, the more infrequent substrings will be added and so considered as potential edges of the conical hull. When the vectors $\mathbf{p}_u$ are not well estimated, they prevent the near-separable NMF algorithm identifying the right edges, leading to potential high errors. Besides the improved performances, being able to work with small basis is a good gain for computation speed and scalability. As expected, applying BW to the output of SepFPA has always improved the solution. This again advocates for the combination of SepPFA and BW. Finally, using two datasets of different sizes shows how BW cannot handle large data sets. The running time of BW between the two datasets has changed from few minutes to at least a day, whereas the one of SepPFA has only slightly increased of one or two seconds, i.e. just the time needed to compute the empirical estimates on a larger dataset.

6. CONCLUSIONS

In this paper, we proposed a new algorithm based on a near-separable NMF and constraint quadratic optimization that can learn in polynomial time a separable PFA from the distribution p it realizes. In addition, even if p is not realized by a separable PFA or is empirically estimated, our algorithm returns a separable PFA that realizes a proper distribution. Then, we empirically demonstrated both, its good performances in comparison to other MoM-based algorithms, and, its scalability in comparison to BW. Finally, experiments have shown that initializing EM with SepPFA has almost always improved the performances of both EM and SepPFA. For future work, one could try to establish the consistency of Algorithm 1 and some finite sample bounds based on the robustness analysis of the near-separable NMF. In addition, extending Algorithm 1 to handle controlled processes such that (Partially Observable) Markov Decision Processes would allows to efficiently learn dialogue systems as proposed in [23].

7. REFERENCES

[1] Mark Johnson, "Why doesn't em find good hmm pos-taggers?," in *Proceedings of EMNLP-CoNLL-07*, 2007.

[2] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[3] Mehryar Mohri, "Finite-state transducers in language and speech processing," *Computational linguistics*, vol. 23, no. 2, pp. 269–311, 1997.

[4] Bernard Merialdo, "Tagging english text with a probabilistic model," *Computational linguistics*, vol. 20, no. 2, pp. 155–171, 1994.

[5] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky, "Tensor decompositions for learning latent variable models," *arXiv preprint arXiv:1210.7559*, 2012.

[6] Raphaël Bailly, Amaury Habrard, and François Denis, "A spectral approach for probabilistic grammatical inference on trees," in *Proc of ALT-10*. Springer, 2010.

[7] Michael Thon and Herbert Jaeger, "Links between multiplicity automata, observable operator models and predictive state representations–a unified learning framework," *Journal of Machine Learning Research*, vol. 16, pp. 103–147, 2015.

[8] Hadrien Glaude, Olivier Pietquin, and Cyrille Enderli, "Subspace identification for predictive state representation by nuclear norm minimization," in *Proceedings of ADPRL-14*, 2014.

[9] Borja Balle, William Hamilton, and Joelle Pineau, "Methods of moments for learning stochastic languages: Unified presentation and empirical comparison," in *Proceedings of ICML-14*, 2014.

[10] Mattias Gybels, François Denis, and Amaury Habrard, "Some improvements of the spectral learning approach for probabilistic grammatical inference," in *Proceedings of ICGI-12*, 2014, vol. 34.

[11] Jack W. Carlyle and Azaria Paz, "Realizations by stochastic finite automata," *Journal of Computer and System Sciences*, vol. 5, no. 1, pp. 26 – 40, 1971.

[12] Borja Balle, *Learning finite-state machines: algorithmic and statistical aspects*, Ph.D. thesis, 2013.

[13] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle H Ungar, "Experiments with spectral learning of latent-variable pcfgs.," in *Proceedings of HLT-NAACL-13*, 2013.

[14] François Denis and Yann Esposito, "On rational stochastic languages," *Fundamenta Informaticae*, vol. 86, no. 1, pp. 41–77, 2008.

[15] Nicolas Gillis, "The Why and How of Nonnegative Matrix Factorization," *ArXiv e-prints*, Jan. 2014.

[16] Nicolas Gillis and Stephen A Vavasis, "Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization," *arXiv preprint arXiv:1310.2273*, 2013.

[17] Yann Esposito, Aurélien Lemay, François Denis, and Pierre Dupont, "Learning probabilistic residual finite state automata," in *Grammatical Inference: Algorithms and Applications*, pp. 77–91. Springer, 2002.

[18] David Pfau, Nicholas Bartlett, and Frank Wood, "Probabilistic deterministic infinite automata," in *Proceedings of NIPS-10*, 2010.

[19] Pierre Dupont, François Denis, and Yann Esposito, "Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms," *Pattern recognition*, vol. 38, no. 9, pp. 1349–1371, 2005.

[20] Sicco Verwer, Rémi Eyraud, and Colin de la Higuera, "Results of the pautomac probabilistic automaton learning competition," *Journal of Machine Learning Research - Proceedings Track*, vol. 21, pp. 243–248, 2012.

[21] Borja Balle, Ariadna Quattoni, and Xavier Carreras, "Local loss optimization in operator models: A new insight into spectral learning," in *Proceedings of ICML-12*, 2012.

[22] Ilya Sutskever, James Martens, and Geoffrey E Hinton, "Generating text with recurrent neural networks," in *Proceedings of ICML-11*, 2011.

[23] Oliver Lemon and Olivier Pietquin, *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces* , Springer, November 2012, 177 pages.